Chapter 18

Running a BASIC–PLUS Program

The BASIC–PLUS/RT–11 program language[1] is a machine-independent programming language that is one of the easiest languages for the beginning programmer to learn. It has both elementary language features that you use to write simple programs, and more advanced operations that let you produce complex and efficient programs. In addition, a special "immediate mode" lets you use BASIC–PLUS like a calculator to obtain instant answers to mathematical problems.

You do not need to understand the BASIC–PLUS language or the way the examples work to successfully perform the exercises in this chapter.

NOTE

In this chapter, the term BASIC–PLUS means BASIC–PLUS/RT–11.

18.1 Developing a BASIC–PLUS Program

BASIC (Beginner's All-purpose Symbolic Instruction Code)–PLUS is conversational in nature. It uses simple English keywords and common mathematical expressions to form easily understood language statements.

You write a BASIC–PLUS program as a series of one or more program lines. You begin each program line with a number that both identifies the line and indicates the order in which the line will be processed. Individual program lines contain one or more BASIC–PLUS language statements that define the operations to be performed.

When you are satisfied with the logic of your BASIC–PLUS source program, you create it as a file. However, unlike your methods under other programming languages, you can create the file under the control of the BASIC–PLUS interpreter. Thus, you can use commands that are part of the BASIC–PLUS language processor to create and edit the program, list it, run it, and save it for later use.

18.2 Using the BASIC–PLUS Interpreter

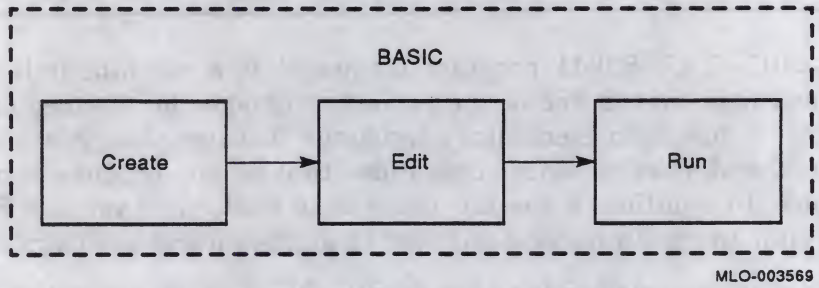
The BASIC–PLUS program, BASIC.SAV, is an interactive interpreter. The interpreter lets you create and execute a program in its entirety or a few lines at a time. The interpreter examines each program language statement, interprets it, and executes it before going on to the next.

If BASIC–PLUS discovers an error that prevents further processing, it calls the error file BASIC.ERR. BASIC–PLUS retrieves and prints on the terminal a message

[1] BASIC–PLUS is a superset of the standard BASIC language developed at Dartmouth College.

informing you of the error condition and stops. You correct the error so that execution can continue past that point and then rerun the program.

Figure 18–1: Functions of the BASIC–PLUS Interpreter



The functions of program creation, editing, processing, and execution are all handled by the BASIC–PLUS interpreter. Some RT–11 systems store the BASIC–PLUS interpreter on a volume apart from the system volume. You can quickly determine whether the BASIC–PLUS interpreter is on your system volume by typing the monitor DIRECTORY command and specifying the BASIC.SAV program. You should also verify that the error message file, BASIC.ERR, resides on your system volume:

```
.DIRECTORY SY:BASIC.* RET
```

In the directory listing that results, if the directory entries for BASIC.SAV and BASIC.ERR are listed on your terminal, then the required BASIC–PLUS files are on your system volume and you are ready to use the interpreter. However, if they did not appear in your listing, then you should copy any missing file to your system volume.

If you are running under the XM monitor, issue the following commands to activate the BASIC–PLUS interpreter:

```
.R BASIC RET
BASIC-PLUS/RT-11 V3.x-xx
Ready
```

If you receive an error message indicating insufficient memory, issue instead the following commands:

```
.VBGEXE RET
Program? BASIC RET
BASIC-PLUS/RT-11 V3.x-xx
Ready
```

If you are not running under the XM monitor, issue the following command to activate the BASIC–PLUS interpreter:

```
.R BASIC RET
BASIC-PLUS/RT-11 V3.x-xx
Ready
```

BASIC–PLUS prints the Ready message to indicate that it is ready to accept a BASIC–PLUS command. Any text that you type that is not preceded by a BASIC–PLUS command is accepted as program (or immediate mode) input. If at any time you wish to return to the monitor command mode, type the EXIT command following the Ready message. The Ready message appears after any BASIC–PLUS execution that is completed or interrupted by pressing CTRL/C twice, or after any BASIC–PLUS wait condition that is terminated by pressing CTRL/C once.

18.2.1 Immediate Mode

Immediate mode lets you use the BASIC–PLUS interpreter like a calculator to obtain immediate answers to arithmetic problems. You enter the appropriate BASIC–PLUS statement keyword and any necessary mathematical formula. When you press RETURN, BASIC–PLUS immediately calculates and prints the results. (Press DELETE or CTRL/U to correct any typing errors.)

```
PRINT (128+75)*3 RET
 609
Ready
```

BASIC–PLUS adds the two numbers in parentheses, multiplies them by 3, and prints the answer. The PRINT statement causes the answer to be printed on the terminal. The following command provides another example:

```
PRINT INT(34.67) RET
 34
Ready
```

The greatest integer less than or equal to 34.67 is printed.

You can combine several statements, including variable names, arithmetic equations, and data. BASIC–PLUS considers all the information, calculates the answer and prints it on the terminal, as illustrated in the following example:

```
A=5 RET
Ready
B=14 RET
Ready
C=.3729 RET
Ready
PRINT "THE HEIGHT IS";A*SIN(C)+B;"METERS" RET
THE HEIGHT IS 15.8216 METERS
Ready
```

The first three statements equate variable names with values; the final statement introduces a formula for calculating a result and prints it.

You can use immediate mode to solve fairly lengthy and complicated mathematical problems. However, immediate mode information is temporary; you cannot save it. You can recall and change statements, if you have previously issued the RT–11 command SET SL ON, to enable the single-line command editor (SL). See Chapter 10 for information about SL.

If your needs are more complex, or if you want to save your statements, you should create a BASIC–PLUS program.

18.2.2 Creating and Editing a BASIC–PLUS Program

You use the BASIC–PLUS NEW command to create a program. The NEW command erases any contents in memory and asks you for a new program name:

```
NEW [RET]
New File Name -
```

Supplying a file name is optional. Do not supply a file name if you are creating a temporary BASIC–PLUS program. Instead, press [RETURN], creating a temporary program named NONAME. Specify a file name if you want to preserve the program. BASIC–PLUS automatically uses the file type .BAS.

To create a BASIC–PLUS program, assign line numbers to the language statements you type on the terminal keyboard. Your program lines are saved in memory and you can transfer program control to specific lines within the program, repeat parts of the program any number of times, store the entire program for later use, and perform other similar operations that are not possible in immediate mode.

Once you have created the program, you use BASIC–PLUS editing commands to list lines, change lines, add and erase lines, and correct typing errors. Create the following example program:

```
NEW [RET]
New File Name - [RET]

Ready
 3 LET T=0 [RET]
 5 FOR I=1 TO 10 [RET]
20 INPUT J [RET]
25 LET T=T+J [RET]
50 NEXT I [RET]
55 PRINT "THE TOTAL IS";T [RET]
88 END [RET]
```

Those program lines are now in memory.

You could at this point preserve this program by issuing the BASIC–PLUS SAVE command. The SAVE command copies the program to the specified storage volume and gives the program the file name and file type that you indicate in the command line. The SAVE command does not alter the current contents of memory. You could, for example, preserve variations of the same program by saving the program at different stages of development to different file names. If you specify no device name or file specification, the SAVE command creates a file with the program name and of type .BAS on the default data (DK) volume.

You can list individual lines by specifying the line number. For example:

```
LIST 5 [RET]

NONAME      23-SEP-89       11:49 AM

5 FOR I=1 TO 10

Ready
```

Note that BASIC–PLUS prints a header line. Since you have not yet assigned a name to your program, BASIC–PLUS assigns it the name NONAME and prints this name, along with the date (which is only correct if previously entered by way of the DATE monitor command) and the time when you use the LIST command. You can omit the header line by using the LISTNH command instead of the LIST command.

You can change the contents of a program line by reentering the line. For example, to change the contents of line 25:

```
LISTNH 25 [RET]

25 LET T=T+J

Ready
25 LET T=T+I [RET]
LISTNH 25 [RET]

25 LET T=T+I

Ready
```

You can display a range of program lines by specifying the line range in the LIST command:

```
LISTNH 50-88 [RET]

50 NEXT I
55 PRINT "THE TOTAL IS";T
88 END

Ready
```

You can also list all the program lines by specifying no line range in the LIST command:

```
LISTNH [RET]

 3 LET T=0
 5 FOR I=1 TO 10
20 INPUT J
25 LET T=T+I
50 NEXT I
55 PRINT "THE TOTAL IS";T
88 END

Ready
```

You can delete a single program line by specifying the line number and pushing [RETURN]:

```
20 [RET]
LISTNH [RET]
```

```
 3 LET T=0
 5 FOR I=1 TO 10
25 LET T=T+I
50 NEXT I
55 PRINT "THE TOTAL IS";T
88 END

Ready
```

You can delete a range of program lines by specifying the line range in the DELETE[1] command:

```
DELETE 25-50 RET
LISTNH RET
 3 LET T=0
 5 FOR I=1 TO 10
20 INPUT J
55 PRINT "THE TOTAL IS";T
88 END

Ready
```

Finally, you can delete the entire program, using the SCRATCH command:

```
SCRATCH RET
LISTNH RET

Ready
```

All program lines are erased from memory.

18.3 BASIC–PLUS Demonstration Program

RT–11 distributes a BASIC–PLUS demonstration program, DEMOB1.BAS, for this chapter. Ensure that DEMOB1.BAS resides on your default data (DK) volume:

```
·DIRECTORY DEMOB1.BAS RET
```

If the directory entry for DEMOB1.BAS is not displayed, copy the file from your software distribution kit to your DK volume. Once you have copied the file to your DK volume, issue the following command to copy the file DEMOB1.BAS to the demonstration program file MATCH.BAS, thereby preserving DEMOB1.BAS in its original state:

```
·COPY DEMOB1.BAS MATCH.BAS RET
```

MATCH.BAS contains the demonstration program 23 Matches.[2] The following is a reproduction of the program:

[1] Do not confuse the BASIC–PLUS DELETE command with the DELETE key on the terminal keyboard.

[2] 23 Matches, *101 BASIC Computer Games*, Maynard, Mass.: Digital Equipment Corporation, 1975.

```
100 REM THE PROGRAM 23 MATCHES
101 REM
110 PRINT "WE BEGIN WITH 23 MATCHES. YOU MOVE FIRST. YOU"
115 PRINT "MAY TAKE 1, 2, OR 3 MATCHES. TYPE YOUR CHOICE"
120 PRINT "FOLLOWED BY A CARRIAGE RETURN. THEN THE COM-"
125 PRINT "PUTER CHOOSES 1, 2, OR 3 MATCHES. YOU CHOOSE"
130 PRINT "AGAIN, AND SO ON. WHOEVER MUST TAKE THE LAST"
135 PRINT "MATCH, LOSES."
140 PRINT \ LET M=23
200 REM THE HUMAN MOVES
201 REM
210 PRINT \ PRINT "THERE ARE NOW";M;"MATCHES."
215 PRINT \ PRINT "HOW MANY DO YOU TAKE";
230 INPUT H
240 IF H>M THEN 510
250 IF H<>INT(H THEN 510
260 IF H<=0 THEN 510
270 IF H>=4 THEN 510
280 LET M=M-H
290 IF M=0 THEN 410
300 REM THE COMPUTER MOVES
301 REM
305 IF M=1 THEN 440
310 LET R=M-4*INT(M/4)
320 IF R<>1 THEN 350
330 LET C=INT(3*RND)+1 \ GO TO 360
350 LET C=(R+3)-4*INT((R+3)/4)
360 LET M=M-C
370 IF M=0 THEN 440
380 PRINT \ PRINT "THE COMPUTER TOOK";C;"....";
390 GO TO 310
400 REM SOMEBODY WON
401 REM
410 PRINT \ PRINT "THE COMPUTER WON." \ GO TO 999
440 PRINT \ PRINT "YOU WON." \ GO TO 999
500 REM BAD INPUT
501 REM
510 PRINT "ENTER ONLY 1, 2, OR 3." \ GO TO 215
999 END
```

As you can see from the first few lines of the listing, this program is a mathematical game where you match your logic against the program logic. The PRINT statements in the program print messages, game instructions, results, and so forth, on the terminal. The REM statements identify comment lines—remarks that provide general information about the program, but that are ignored by BASIC–PLUS during processing. The INPUT statement in line 230 lets you supply data to the terminal. Depending on the value you enter, program control transfers to various other parts of the program. For example, if you type an invalid value, program control skips ahead to a PRINT statement in line 510 informing you of your mistake and then returns to line 215 to ask for a value again. The mathematical algorithms of this program are in lines 310 through 350, which determine the number of matches the computer will select based on your choice.

18.4 Running a BASIC–PLUS Program

Once you have ensured that MATCH.BAS resides on your DK volume, you read MATCH.BAS into memory, using the BASIC–PLUS OLD command:

```
OLD RET
Old file name-MATCH.BAS RET
Error - Invalid expression at line 250

Ready
```

As BASIC–PLUS reads MATCH.BAS into memory, it checks the syntax of MATCH.BAS and finds an error on program line 250. Although an error is found, the program is in memory. You can now display the program line that contains an error:

```
LISTNH 250 RET

?250 IF H<>INT(H THEN 510)

Ready
```

Note that the right parenthesis is missing after the second H in the line. Correct the line by retyping it:

```
250 IF H<>INT(H) THEN 510 RET
```

You are now ready to run the program. The BASIC–PLUS RUN command initiates program execution. This command prints a header that includes the program name, date, and time. If you want to omit the header line, type the RUNNH command instead:

```
RUNNH RET
```

You see this text print on your terminal:

```
WE BEGIN WITH 23 MATCHES. YOU MOVE FIRST. YOU
MAY TAKE 1, 2, OR 3 MATCHES. TYPE YOUR CHOICE
FOLLOWED BY A CARRIAGE RETURN. THEN THE COM-
PUTER CHOOSES 1, 2, OR 3 MATCHES. YOU CHOOSE
AGAIN, AND SO ON. WHOEVER MUST TAKE THE LAST
MATCH, LOSES.

THERE ARE NOW 23 MATCHES.

HOW MANY DO YOU TAKE?
```

When the program pauses and asks you a question, you must supply data, in this case a 1, 2, or 3. Enter your choice (represented here by *n*):

```
n RET

Ready
```

BASIC–PLUS begins processing and enters an infinite loop, a series of commands that it repeats endlessly. After several lines have printed, press CTRL/C CTRL/C; this interrupts execution and returns control to BASIC–PLUS command mode, as follows:

```
n RET
```

```
THE COMPUTER TOOK 1 ....
THE COMPUTER TOOK 1 ....
THE COMPUTER TOOK 3 ....
THE COMPUTER TOOK 2 ....
THE COMPUTER TOOK 2 ....
THE COMPUTER TOOK 3 ....
THE COMPUTER TOOK 1 ....
THE COMPUTER TOOK 1 ....
THE COMPUTER TOOK 3 ....
THE COMPUTER TOOK 1 ....
THE COMPUTER TOOK 3 ....
CTRL/C  CTRL/C

Ready
```

An infinite loop is a programming logic error. However, since the error does not prevent processing, BASIC–PLUS does not print an error message. Instead BASIC–PLUS is caught in a loop of instructions and executes them endlessly. This particular loop is obvious because it prints a line of text; other kinds of loops may not be so evident. At this point, you must examine the program logic to determine why these instructions are being repeated.

Look at the program listing. The problem in this case is at line 390. That line instructs program control to return to line 310; therefore lines 310 through 390 are repeated endlessly without ever obtaining your next value choice. Program control should really return to line 210. Correct line 390 as follows by retyping it:

```
390 GO TO 210 RET
```

Now, you are ready to run the program again. This time the entire program should execute without error. Enter your value choices when requested. (A hint to playing the game: your first value choice determines whether you can win; if your first choice is wrong, the program has the advantage throughout.) A sample run follows.

```
RUNNH RET

WE BEGIN WITH 23 MATCHES. YOU MOVE FIRST. YOU
MAY TAKE 1, 2, OR 3 MATCHES. TYPE YOUR CHOICE
FOLLOWED BY A CARRIAGE RETURN. THEN THE COM-
PUTER CHOOSES 1, 2, OR 3 MATCHES. YOU CHOOSE
AGAIN, AND SO ON. WHOEVER MUST TAKE THE LAST
MATCH, LOSES.

THERE ARE NOW 23 MATCHES.

HOW MANY DO YOU TAKE? 1 RET

THE COMPUTER TOOK 1 ....
THERE ARE NOW 21 MATCHES.

HOW MANY DO YOU TAKE? 1 RET

THE COMPUTER TOOK 3 ....
THERE ARE NOW 17 MATCHES.

HOW MANY DO YOU TAKE? 2 RET

THE COMPUTER TOOK 2 ....
THERE ARE NOW 13 MATCHES.

HOW MANY DO YOU TAKE? 1 RET

THE COMPUTER TOOK 3 ....
THERE ARE NOW 9 MATCHES.
```

```
HOW MANY DO YOU TAKE? 1 RET

THE COMPUTER TOOK 3 ....
THERE ARE NOW 5 MATCHES.

HOW MANY DO YOU TAKE? 3 RET

THE COMPUTER TOOK 1 ....
THERE ARE NOW 1 MATCHES.

HOW MANY DO YOU TAKE? 0 RET
ENTER ONLY 1, 2, OR 3.

HOW MANY DO YOU TAKE? 1 RET

THE COMPUTER WON.

Ready
```

You can repeat the program as often as you like, using the BASIC–PLUS commands RUN or RUNNH.

You do not now want to exit from BASIC–PLUS as you would not preserve the corrected MATCH program. If the original version of the MATCH program resided in memory, you would preserve it with the SAVE command. However, because MATCH.BAS already exists on a storage (the DK) volume, you preserve the corrected version with the REPLACE command:

```
REPLACE RET

Ready
```

The current version of MATCH.BAS is written to DK and the MATCH program remains in memory.

18.5 Using the KED Editor with BASIC–PLUS Programs

BASIC–PLUS provides commands you can use to perform file operations outside the BASIC–PLUS interpreter. You can create and edit BASIC–PLUS programs by using the KED editor.

The OLD command reads an existing BASIC–PLUS program from an ASCII text file into computer memory. As this command reads in the program, BASIC–PLUS converts the program from ASCII text format to an intermediate format used by BASIC–PLUS.

The SAVE and REPLACE commands copy a BASIC–PLUS program from computer memory to a storage volume. As these commands copy the program, they convert it from the intermediate format used by BASIC–PLUS to ASCII text format.

Thus, you can, if you prefer, use the KED editor to create and edit BASIC–PLUS programs, since the editor also uses ASCII text format. However, many users would rather use BASIC–PLUS to create and edit a BASIC–PLUS program, since they can then run the program, reedit it, rerun it, and save the new version—all in BASIC–PLUS command mode—rather than perform the several corresponding monitor commands.

18.6 Preserving the Intermediate Translated Program

When you create, read into memory, or edit a BASIC–PLUS program, the interpreter creates an intermediate translation of each command line. This intermediate translation process lets you run a BASIC–PLUS program as soon as you create it or read it into memory. Once you have finalized a BASIC–PLUS program, you can preserve the intermediate translation in a file by issuing the BASIC–PLUS COMPILE command. Preserving the translated program version eliminates the need for BASIC–PLUS to translate the ASCII source program each time you run it.

A translated BASIC–PLUS program has the file type .BAC, and BASIC–PLUS looks first for the .BAC file type when you issue the command to run (RUN or RUNNH) a program name. BASIC–PLUS looks for the ASCII source (.BAS) version of a program only when it cannot find the intermediate translated (.BAC) version.

Although translation is most effective with larger BASIC–PLUS programs, for practice, preserve the intermediate translation of the program MATCH, by creating MATCH.BAC on your DK device:

```
COMPILE MATCH RET
Ready
```

You cannot use the OLD command to read a translated (.BAC) program into memory; you must run the program, using the RUN or RUNNH command. Also, the .BAC version of a program is not a source file and cannot be edited. You should, therefore, always preserve the .BAS version of each BASIC–PLUS program in case you want to edit the program in the future.

18.7 Stopping the BASIC–PLUS Interpreter

You stop the BASIC–PLUS interpreter and return to the RT–11 monitor by issuing the EXIT command:

```
EXIT RET
.
```

18.8 Summary of Commands

The following BASIC–PLUS commands are described and illustrated in this chapter. If the meaning of any command is unclear, review the example in this chapter that illustrates the command.

line

Erases the indicated program lines.

COMPILE

Copies the translated version of the BASIC–PLUS program currently in memory to the specified volume (default volume is DK).

? missing special feature

CONTINUE → continues after an issued ^C.

CTRL/C

Under control of BASIC–PLUS only, interrupts execution of the BASIC–PLUS program and returns control to BASIC–PLUS command mode.

DELETE *line #*

Erases the indicated program lines.

EXIT

KILL "[x,y]*.*" deletes files.

Returns control to monitor command mode (only when using BASIC–PLUS).

LIST

Lists the entire program and prints a header that includes the program name, date, and time.

LIST *line #*

Lists the indicated lines and prints a header that includes the program name, date, and time.

LISTNH

Lists the entire program but does not print a header.

LISTNH line #

Lists the indicated lines but does not print a header.

NEW

Creates a new BASIC–PLUS program and assigns the indicated file name.

OLD

Copies into memory an existing BASIC–PLUS program (for use under BASIC–PLUS).

RENAME → makes a new copy renames program in memory.

REPLACE

Copies the BASIC–PLUS program currently in memory to the indicated storage volume and replaces the version that already exists on that volume.

RUN

Executes the BASIC–PLUS program currently in memory; prints a header line that includes the program name, date, and version number.

RUNNH

Executes the BASIC–PLUS program currently in memory; omits the header line.

SAVE

Copies the BASIC–PLUS program currently in memory to the indicated storage volume.

[1,2] of [001002]